

FreeRADIUS

The world's most popular RADIUS Server

FreeRADIUS 新手入门

Akagi201

目录

FreeRADIUS Beginner's Guide

捐款连接

- 本人已经不做 RADIUS 这块东西了. 由于有大量的朋友给我邮件来索取和督促我来继续翻译完这边书, 所以, 放个捐款连接吧.
- 规则: 每收到 10 块钱, 更新一章.



Outline

排版规范

- 中文文案排版指北
- 有一点特别的地方是, 我本人的全用英文半角标点符号, 所以, 希望大家也用 "半角标点+空格" 的方式.

Git 协作方式

- CONTRIBUTING

Gitbook 指南

- 安装命令行工具: `npm install -g gitbook-cli` 建议使用最新 3.0 以上版本.
- build 前先安装 `book.json` 中配置的插件, `gitbook install` .
- 本地编译: `gitbook build` 会将 build 后网页文件放到 `_book` 目录.
- 本地查看: `gitbook serve` 默认会进行 build, 然后 serve 一个网页提供给本地浏览器访问.

LICENSE

- LICENSE

FreeRADIUS新手入门包含了大量实际的练习来帮助你学会从基础的安装到更高级的配置像集成LDAP和Active Directory的一切.这本书将会帮助你理解在FreeRADIUS里面的authentication, authorization和accounting使用现在最流行的linux发行版. 使用realms和fail-over配置的更大规模的部署也在tips里面涵盖到了. 每个章节最后有一个课堂测试来验证你的理解.

这本书涵盖了哪些内容

这本书可以被划分为3个部分:

1. 介绍和安装(第一章到第三章)
2. FreeRADIUS的AAA功能(第四章到第七章)
3. 高级话题(第八章到第十三章)

让我们来看看每一章的内容:

第一章, 介绍AAA和RADIUS, 介绍FreeRADIUS和RADIUS协议. 他着重介绍了一些关键的RADIUS概念, 帮助用户避免常见的误解.

第二章, 安装, 描述在流行的linux发行版上如何从源码编译和安装FreeRADIUS. 也涵盖了使用流行的linux发行版的预编译的FreeRADIUS的安装包来安装. Ubuntu, SUSE, 和 CentOS将会被使用来确保广泛的覆盖.

第三章, 开始使用FreeRADIUS, 简要的介绍FreeRADIUS的许多组件. 也讨论了处理一个基本authentication请求的过程.

第四章, Authentication, 教授authentication方法和他们如何工作. EAP(Extensible Authentication Protocol)在后面一个专门的章节讲解.

第五章, 用户名和密码的来源, 涵盖许多用户名/密码组合可以存储的地方. 他显示了哪些模块被包含并且怎样配置FreeRADIUS来实现这些存储.

第六章, Accounting, 讨论accounting的需求和记录accounting数据的可选项. 也讨论了实现一个policy, 包含限制sessions和/或时间和/或数据.

第七章, Authorization, 讨论了authorization的许多方面, 包括unlang的使用.

第八章, 虚拟服务器, 讨论了虚拟服务器的许多方面和他们可能被潜在地使用的地方.

第九章, 模块, 讨论了FreeRADIUS使用的许多模块和如何配置某个模块的multiple instances.

第十章, EAP, 一个专门讨论EAP的章节, 是一个一站式EAP(802.11x和WiFi)的章节.

第十一章, 字典, 介绍字典, 用来映射管理员看到和使用的名字, 到RADIUS协议使用的数字.

第十二章, 漫游和代理, 处理RADIUS协议, 允许authorization和accounting请求的代理. 这个让漫游成为可能. 这章包含了关于FreeRADIUS中使用代理的许多方面.

第十三章, 排查问题, 解答许多常见问题, 给出例子如何去搜索, 和如何修复问题.

阅读这本书的前提

你需要熟悉linux和对TCP/IP有扎实的理解. 不需要提前知道RADIUS和FreeRADIUS.

为了最好的实践练习, 你需要一个干净安装的Ubuntu, SUSE或者CentOS.

这本书是为了谁写的?

如果你是一个IPS或者网络管理者, 需要跟踪和控制网络使用, 那么这本书是为你准备的.

读者反馈

来自我们的读者的反馈永远是欢迎的. 让我们知道你对这本书的看法 -- 无论你是否喜欢. 读者的反馈对我们来说是重要的. 读者的反馈是重要的对于我们来写出你最能好好利用的话题.

为了给我发送反馈, 简单低发送一封邮件到feedback@packtpub.com, 并且提到书名在你的邮件的主题中.

如果有本书你需要并且想要看到我们发布, 请发送给我们通过http://packtpub.com网站上的 `SUGGEST A TITLE` 或者发送邮件到suggest@packtpub.com

如果有一个话题你有经验并且感兴趣, 想要写或者贡献一本书, 可以看网站http://packtpub.com/authors上的 `author guide` .

客户支持

由于你是Packet书的骄傲的拥有者, 我们有许多东西来帮会组你来充分利用你的购买.

下载这本书的样例程序

你可以在网站http://www.PacktPub.com上通过你的购买账号下载所有你购买的书的样例程序. 如果你在其他地方购买, 你可以访问http://www.PacktPub.com/support并且注册来让邮件直接发送给你.

勘误表

尽管我们已经已经很小心的确保我们内容的准确性, 错误也会发生. 如果你在我们其中的一本书中发现错误--也许是文字中错误或者代码中错误--我们将会感激如果你愿意报告给我们. 通过这样做, 你可以让其他读者避免挫折并且帮助我们改进这本书的子版本. 如果你发现任何错误, 请访问http://www.packtpub.com/support来报告他们, 选择你的书, 点击 `errata submission`

`form` 链接，并且输入你的错误的细节。一旦你的错误被验证，你的提交将会被接受并且勘误将会被上传到我们的网站，或者被添加到任何现有的勘误列表，在那个书名的勘误部分。任何现有的看屋可以通过选择你的书名从http://www.packtpub.com/support来查看。

版权侵权

互联网上内容的版权侵权是一个一直存在的问题，包括任何媒介。在Packt，我们会认真保护我们的版权和许可证。如果你偶然发现任何我们作品的非法拷贝，以任何形式，在互联网上，请立即提供给我们地址或者网址，这样我们可以追踪补救。

请通过copyright@packtpub.com联系我们，附带一个怀疑侵权的链接。

我们欣赏你保护我们作者的帮助，这样我们能够给你有价值的内容。

问题

你可以通过questions@packtpub.com联系我们，如果你关于这本书的任何方面有问题，这样我们将会尽我们最大努力查出他。

我很荣幸展示给你一个FreeRADIUS入门. 这本书将会帮助你部署一个可靠的, 稳定的, 和可扩展的RADIUS服务器在你的环境.

这个章节是用来介绍RADIUS和FreeRADIUS. 我们将会介绍许多理论并且推荐你要留意他. 这将会提供给你关于RADIUS协议工作原理的一个好的基础, 并且将会在后续的章节中提供很多帮助.

在这一章我们将会

- 了解什么是AAA, 和我们为什么需要他.
- 学习RADIUS从哪开始, 和他为什么如今如此有意义.
- 了解为什么FreeRADIUS作为一个RADIUS服务器确实表现突出.
- 理解AAA, RADIUS和FreeRADIUS之间的关系.

用户通过许多设备获取数据网络和网络资源的权限. 这个发生在一个大范围的硬件上. 以太网交换机, Wi-Fi access points, 和VPN服务器都提供网络访问.

当这些设备用来控制网络访问, 例如一个带有WPA2企业版安全实现的WiFi access point或者一个带有802.1x(EAP)的基于端口的认证的以太网交换机, 他们被称做一个NAS(Network Access Server).

所有这些设备需要进行一些形式的控制来确保合适的安全和使用. 这个要求通常被描述为AAA(Authentication, Authorization和Accounting). AAA有时也被称为"三A框架"(Triple A Framework). AAA是一个高层次的架构模型, 可以被用各种方法实现.

AAA是通过许多RFC来指定的. 一般的AAA架构是在RFC2903上指定的. 也有许多RFC涵盖不同的AAA方面.

Authentication

Authentication通常是考虑的第一步, 为了获取网络访问和他提供的服务. 这是一个过程用来确认是否Alice提供的credential是有效的. 提供credential的最常见的方式是通过用户名和密码. 其他方式像one-time token, certificate, PIN numbers, 或者甚至biometric scanning也可以使用.

在成功authentication之后一个session被初始化. 这个session持续直到网络连接被终止掉.

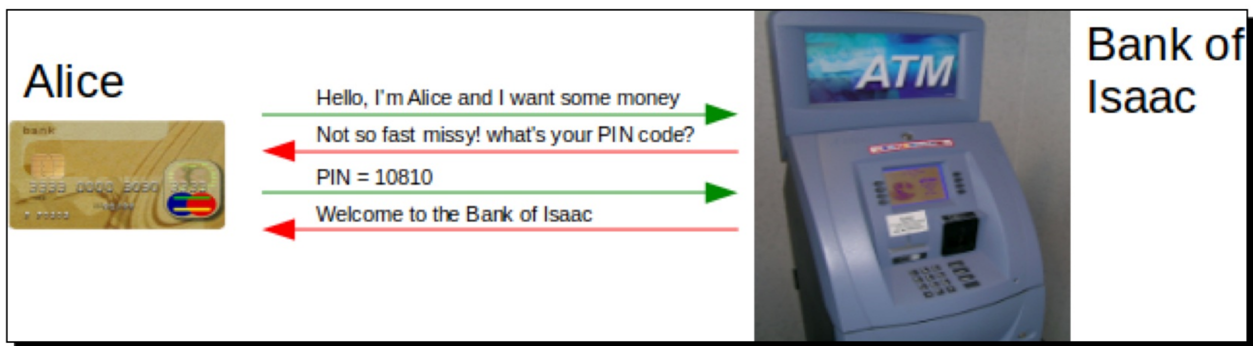
Alice是谁?

Alice和Bob是占位名. 事实上, 有一个完整的角色集合, 每一个代表一个指定的角色. 我们将会使用下面的占位名:

- Alice: 一个想要访问我们网络的用户.
- Bob: 另一个想要访问我们网络的用户.
- Isaac: ISP(Internet Service Provider)/我们的网络

更多的介绍: http://en.wikipedia.org/wiki/Alice_and_Bob

下面的图片阐述了一个authentication过程通过使用常见的活动, 从ATM机中取钱. 这个实质上让你获取访问银行网络的权限(尽管他非常有限制).



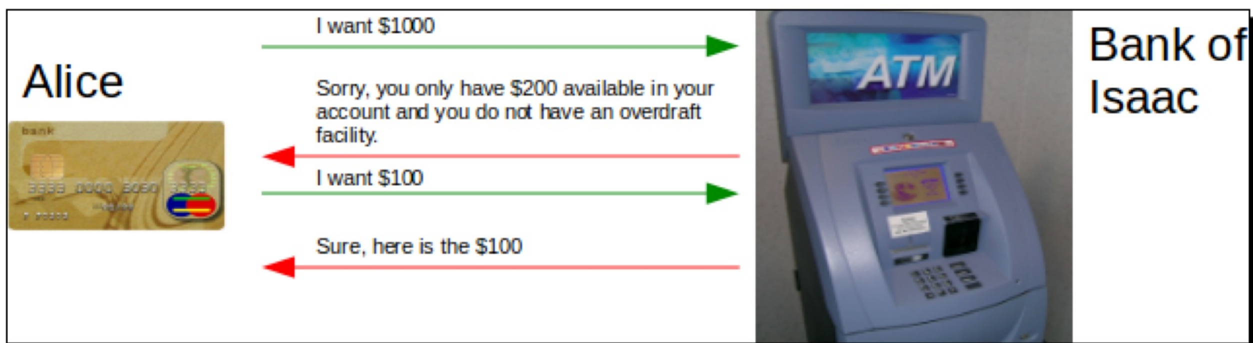
Authorization

Authorization是一种Isaac控制资源使用的方式. 在Alice已经authenticate了她自己之后, Isaac可以强加某些限制或者授予某些权利. Isaac可以例如检查Alice从哪台设备访问网络和基于这点来做出决定. 他可以限制Alice可以拥有的打开的session的个数, 给她一个预先决定的IP地址, 只允许某些流量通过, 或者甚至执行QoS基于一个SLA(Service Level Agreement, 服务级别协议).

Authorization通常包含逻辑. 如果Alice是学生组的一部分那么在工作时间没有网络访问权限. 如果Bob通过一个captive portal访问网络, 那么一个带宽限制被强加来阻止他超量使用网络连接.

逻辑可以基于许多事情. Authorization决定例如可以基于组成员或者你连接的NAS或者甚至你访问资源是星期几.

如果我们举前面ATM的例子, 我们可以看到如果Alice没有一个透支机制, 她会被限制她可以取出的钱的数量.

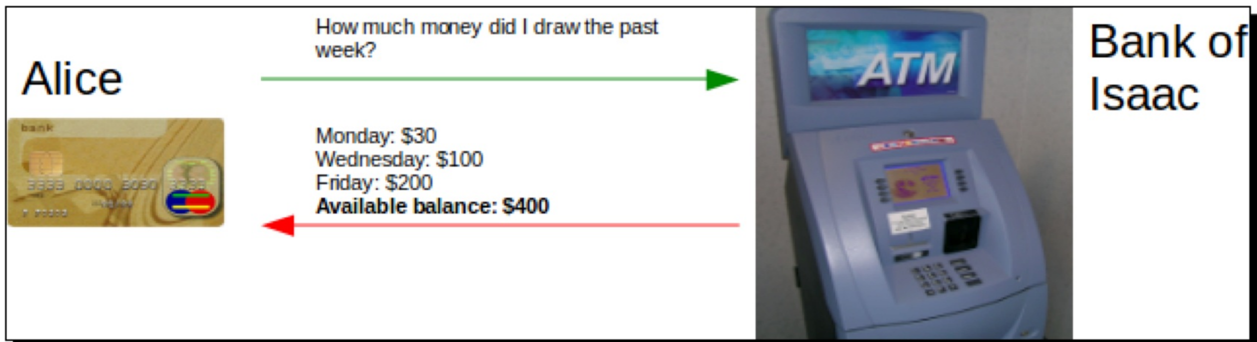


Accounting

Accounting是一种测量资源使用量的方式. 在Isaac已经确定谁是Alice并且在建立的session上强加合适的控制, 他也可以测量她的使用量. Accounting是测量使用量的不间断的过程.

这允许Isaac来跟踪Alice花费了多少时间和资源在一个建立的session上. 获取accounting数据允许Isaac来对Alice的资源进行计费, 趋势分析, 和行为监视.

当Alice想要检查她的使用量和可取出的钱, ATM提供这个功能. Isaac银行也可以监视他的账户和发现是否她经常在月底前花光. 他们然后可以提供给他一个透支机制.



RADIUS是一种协议用来提供在TCP/IP网络上的AAA. 下一节将会继续更多关于RADIUS协议的内容.

RADIUS是Remote Access Dial In User Service的缩写. RADIUS是一个AAA解决方案的一部分, 由Livingston企业发布到Merit Neowrk在1991年. Merit Network是一个非盈利互联网提供商, 需要一种创新的方式来管理通过他们的网络dial-in access to various Points-Of-Presence (POPs).

Livingston企业提供的解决方案有一个中心用户存储用来authentication. 这个可以被许多RAS(Radio Access Station)(dial-in)服务器使用. Authorization和accounting也可以在AAA满足的条件下完成. 另外一个Livingston解决方案的关键方面是包括代理来支持扩展.

RADIUS协议随后在1997年作为RFC发布, 一些修改被提交, 并且现在我们有了RFC2865涵盖了RADIUS协议, 还有RFC2866涵盖了RADIUS accounting. 也有许多额外的RFC涵盖某些RADIUS方面的增强. 让RFC工作从任何个人或供应商到在他们的设备或软件上实现RADIUS协议. 这个导致RADIUS的广泛采用来处理在TCP/IP网络上的AAA. 你将会发现术语RADIUS既表示RADIUS协议也表示完整的RADIUS C/S 系统. 含义应该根据上下文可以明确.

支持RADIUS协议和标准成为NAS供应商的事实标准需求. RADIUS在广泛的地方被使用, 从拥有百万级用户的蜂窝移动网络提供商到小的WISP创业公司提供local neighborhood with Internet 到实现使用802.1x来围栏他们的网路的NAC(Network Access Control)企业网络. RADIUS在所有这些地方和更多的地方可以被发现.

ISP和网络管理员们应该熟悉RADIUS, 由于他被许多控制访问TCP/IP网络的设备使用. 下面是一些例子:

- 一个带有VPN服务器的防火墙可以使用RADIUS.
- 带有WPA-2企业加密的Wi-Fi AP包含RADIUS.
- 当Alice通过DSL连接一个现有的电信设施时, 电信设备将会使用RADIUS协议来联系Isaac的RADIUS服务器为了决定她是否可以通过DSL(代理)获得网络访问权限.

下一节将会总结在RFC2865上指定的RADIUS协议.

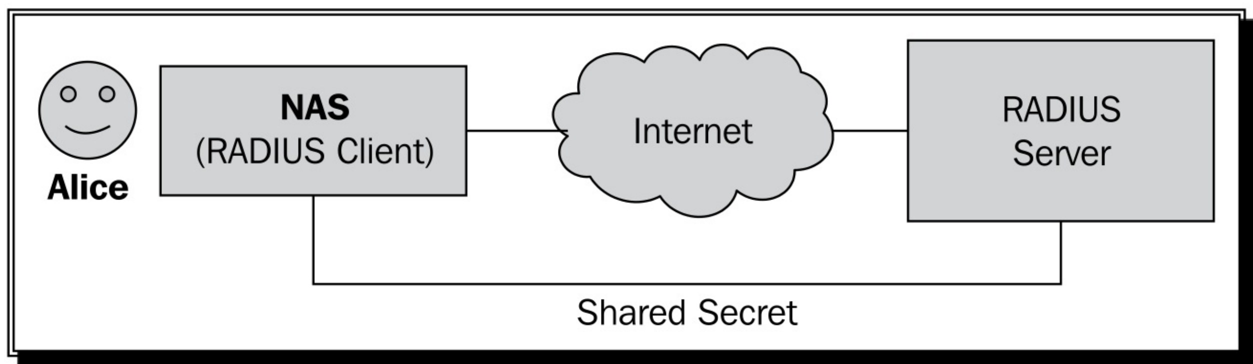
RADIUS协议[RFC2865]

这一节在技术层面探索发布在RFC2865上的RADIUS协议. RADIUS accounting是排除在外的. 这个是发布在RFC2866并且有个关于他自己的章节会详细探索.

RADIUS协议是一个C/S协议, 利用UDP来通信. 使用UDP而不是TCP暗示通信不是严格on state. 在客户端和服务器端的一个典型的数据流有一个来自客户端的单一请求, 紧接着一个来自服务器端的单一回复构成. 这让RADIUS成为一个非常轻量级的协议并且对在低网速环境下的效率有帮助.

在客户端和服务器端之间可以建立成功的通信之前, 每一端必须定义一个shared secret. 这个是用来authenticate客户端.

一个NAS作为一个RADIUS客户端. 因此当你读到关于一个RADIUS客户端, 他表示一个NAS.



RADIUS包有一个指定的格式，定义在RFC里面。在一个RADIUS包中，两个关键的组件是：

- code: 表示包的类型。
- attribute: 携带RADIUS使用的必要数据。

让我们来研究一下RADIUS数据报的组成。

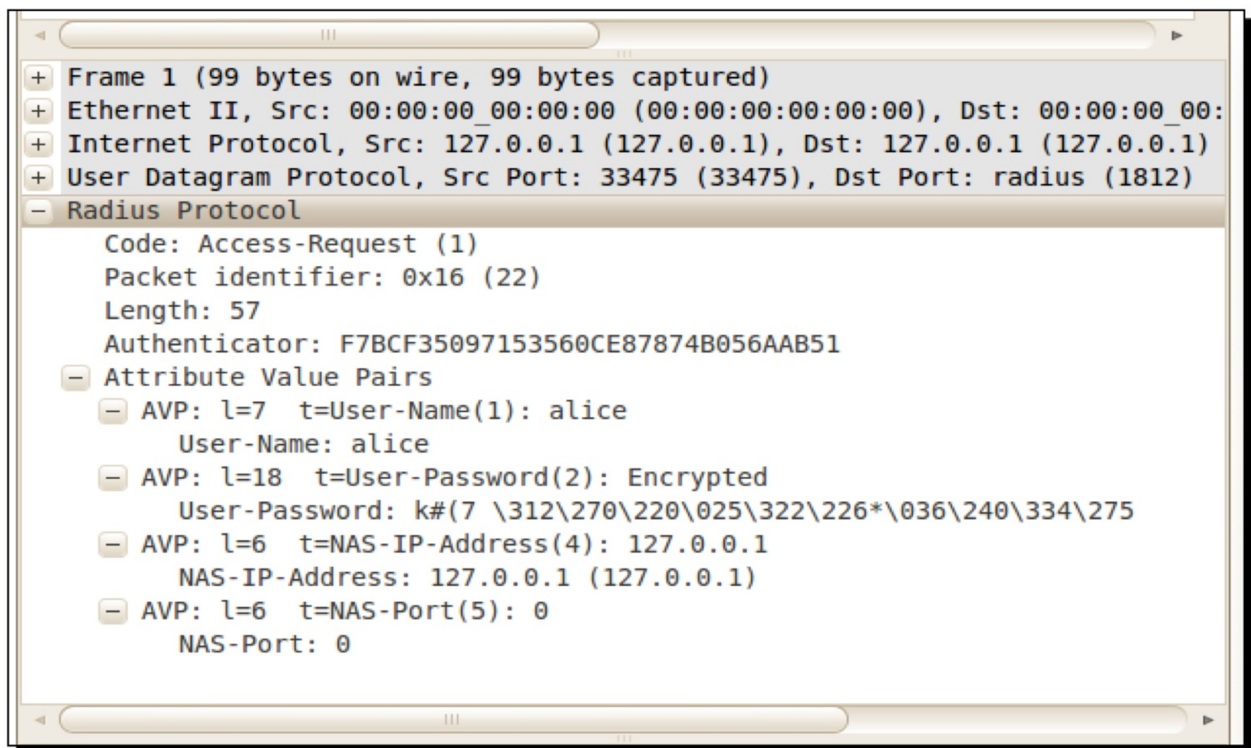
数据包

了解一个RADIUS数据包的格式将会大大帮助理解RADIUS协议。让我们更细致的查看RADIUS数据包。我们将会看一个简单的authentication请求。一个客户端发送一个Access-Request包到服务器。服务器返回一个Access-Accept包来表示成功。

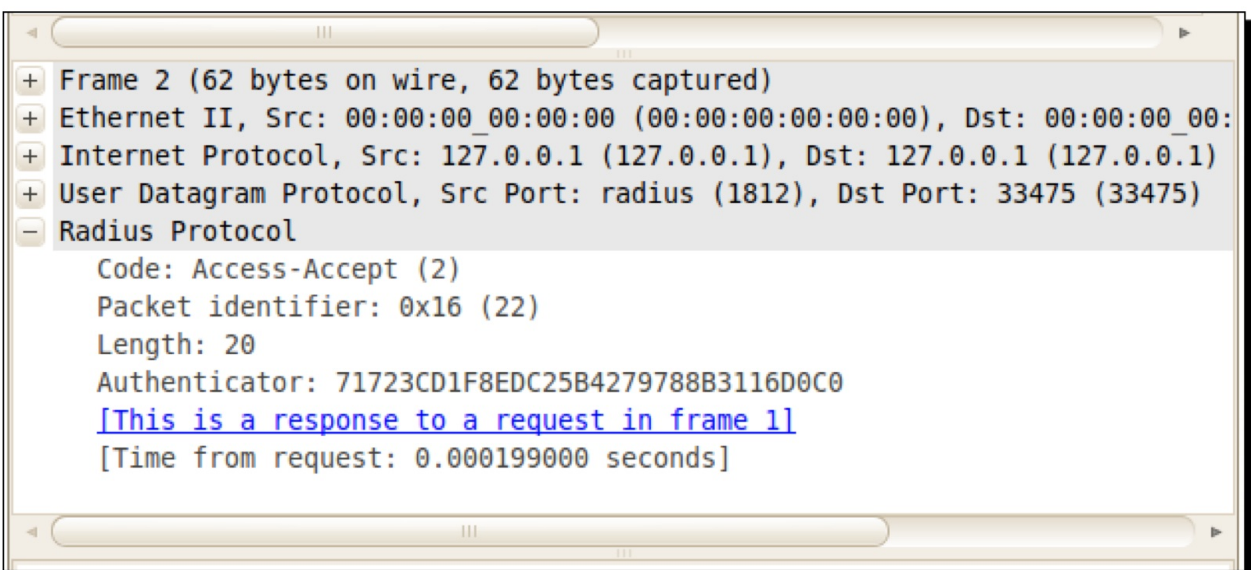
显示在这里的RADIUS数据包只是一个UDP数据包的payload。UDP和IP协议的讨论已经超出这本书的范围了。

屏幕截图是捕捉在RADIUS客户端和服务器端之间的流量得到的。这里的屏幕截图是一个简单的Authentication请求发送到一个RADIUS服务器的结果。

下面的截图显示来自RADIUS客户端的Access-Request包。



下面的截图显示RADIUS服务器返回给这个请求一个Access-Accept数据包.



让我们来研究一下这些数据包.

Code

每一个数据包由一个code来识别. 这个域是一个字节. 这个code的值决定这个包的某些特性和需求. 下面的表格可以被用作一个索引来列出当前RADIUS数据包的定义的code.

RADIUS code(decimal)	Packet type	Sent by
1	Access-Request	NAS
2	Access-Accept	RADIUS server
3	Access-Reject	RADIUS server
4	Accounting-Request	NAS
5	Accounting-Response	RADIUS server
11	Access-Challenge	RADIUS server
12	Status-Server(Experimental)	
13	Status-Client(Experimental)	
255	Reserved	

当跟RADIUS工作时，了解这些code是有利的.

Identifier

每个包的第二个字节包含一个唯一的标识符.他是由客户端生成的，并且用做辅助来匹配请求和回复. RADIUS数据包是通过无连接的UDP传输的.这样需要RADIUS来实现他自己的算法来提交从客户端的重试请求.当一个客户端重发一个请求给服务器，数据包的identifier将会保持不变.服务器将会回复请求通过匹配在回复数据包中的identifier.

Length

这个是数据包的第3，第4个byte.他表示一直到哪里在数据包中是有用的数据.在这个边界之外的字节被认为是填充和被默默地忽略.

Authenticator

这个域占用2个byte，根据包是来自客户端还是服务器端有不同的格式.他也依赖包的类型，例如，Access-Request 或者 Accounting-Request.如果他是一个请求，这个域作为一个Redquest Authenticator.如果他是一个回复，这个域作为一个Response Authenticator.

一个Request Authenticator的值一个不会重复的随机数.Response Authenticator的值是在回复包的许多域的MD5 哈希值，带有客户端和服务器端之间的共享密钥(shared secret).

如果请求包含User-Password attribute，那么这个attribute的值会被加密.这个加密的值通常是把共享的密钥(shared secret)结合authenticator生成MD5哈希值，然后与用户的密码进行异或生成的.这就是为什么共享密钥(shared secret)要在客户端和服务端保持一致，为了解密用户的

密码.

Attributes

RADIUS包的其余部分包含0或者更多attribute, 作为AVP(Attribute Value Pairs). 这些AVPs的结尾是由包的length域指定的.

Conclusion

RADIUS包是通过UDP传输. code域表示RADIUS包的类型. attribute是用来提供指定的信息用于authentication, authorization 和 accounting. 例如为了authenticate一个用户, User-Name和User-Password AVPs将会和一些其他的attribute一起被包含在Access-Request包中.

AVPs

AVPs是RADIUS协议的苦力. AVPs可以被分类为check或者reply attribute. check attribute是从客户端发到服务器端. reply attribute是从服务器端发到客户端.

attribute 作为信息的载体在客户端和服务器端之间服务. 他们被客户端使用来提供他们自身的信息和连接到他上的用户. 他们也被使用当服务器端回复客户端时. 客户端然后可以使用这个回复来控制用户的连接基于在服务器端回复接收到的AVPs.

下一节将会描述一个AVP的格式.

Type

AVP的第一个byte是type域. 这个域的数值是和一个attribute名称联系在一起的, 所以我们人类也可以理解. 这些attribute和数值的关系是由IANA(http://www.iana.org)控制的. attribute的名称通常是有描述性的, 例如 User-Name(1), User-Password(2), 或者 NAS-IP-Address(4).

RADIUS也允许扩展协议. attribute类型26(叫做Vendor-Specific)允许这样做. Vendor-Specific的attribute的值可以反过来包含由一个vendor管理的Vendor Specific Attributes(VSAs).

Length

length域是AVP的第二个byte. 这个与RADIUS包中的length一个含义, 用来表示AVP的长度. 这个方法允许AVPs带有不同大小的值由于length域将会标记AVP的结尾.

Value

AVP的value可以在大小上不同. 这个value域可以是0或者更多byte. 这个value域可以包含一下数据类型中的一个: test, string, address, integer或者time.

Text和string可以多达253byte. Address, integer和time是4个byte大小.

如果我们取出一个请求包中的NAS-IP-Address AVP, 我们会看到长度是6 byte. 一个byte表示类型, 一个byte表示长度, 然后4个byte表示IP地址, 总共6byte.

下一节将会讨论Vendor-Specific Attribute, 是标准的AVPs的扩展.

Vendor-Specific Attributes(VSAs)

VSAs允许vendor来定义他们自己的attribute. attribute定义的格式基本上是普通的AVPs带有额外的一个vendor域名. VSAs是通过AVP类型26发送. 这表示VSAs是AVPs的一个扩展, 并且包含在AVPs之内.

这让RADIUS非常灵活, 并且允许一个vendor来创建扩展来自定义他们的RADIUS实现. 例如CoovaChilli有一个VSA attribute叫做ChilliSpot-Max-Total-Octets. 当CoovaChilli客户端从RADIUS服务器接收到这个attribute的回复时, 他使用这个值来限制通过captive portal的数据.

NAS将会默默地忽略任何他不支持的VSAs. 一些vendors发布了他们的VSAs, 但是这不是需要的. 其他仅仅列出他们在网站或者文档里. 这个然后可以被用来决定他们设备实现RADIUS的能力.

Proxying 和 realms

RADIUS协议允许扩展. Proxying允许一个RADIUS服务器来作为一个客户端连接到另一个RADIUS服务器. 这个最终形成一个链条.

在proxying上的讨论也包含realms. Realms是名字用来组织用户和形成用户名的一部分. 一个用户名区分于realm name带有一个指定的分隔符. realm name可以是用户名加上前缀或者后缀. 今天的流行的标准使用玉米作为后缀, 并且用@分割, 例如: `alice@freeradius.org` . 这个然而只是一个约定. realm可以是任何名字, 分隔符也可以是任何符号. windows用户通常用一个前缀指定域名, 用 `\` 作为分隔符, 例如: `my_domain\alice` .

当RADIUS服务器接收到一个请求带有用户名包含一个realm时, 他决定是否处理这个请求或者转发这个请求给另一个RADIUS服务器被设计用来处理这个指定的realm. 这将会需要第二台RADIUS服务器应该有转发RADIUS服务器作为一个客户端定义, 并且他们也有一个共享的密钥(shared secret).

RADIUS服务器端

RADIUS协议是基于C/S架构. RADIUS服务器将会监听UDP端口1812和1813. 1812端口是用于authentication. 这将会包含Access-Request, Access-Accept, Access-Reject, 和Access-Challenge包. 1813端口用于accounting. 这个将会包含Accounting-Request和Accounting-Response包.

一个客户端和服务器需要一个共享的密钥(shared secret)为了加密和解密在RADIUS包中的某些域.

RADIUS客户端

RADIUS客户端通常提供访问TCP/IP网络. 客户端作为一个RADIUS服务器和一个想要网络权限的用户/设备之间的中间人.

RADIUS的proxying功能也允许一个RADIUS服务器成为另一台RADIUS服务器的客户端, 将会最终形成一个链条.

来自RADIUS服务器的反馈不仅仅决定是否一个用户允许访问网络(authentication), 也可以引导客户端来强加某些限制在用户上(authorization). 限制的例子有在session上的一个时间限制或者限制连接速度.

是否强加推荐的对于用户sessiion的调整取决于客户端. 域名RADIUS协议的无状态的本质, 对于RADIUS服务器没有方法来指导是否客户单在执行推荐的限制.

Accounting定义在一个单独的RFC. 下一节将会总结在RFC2866里面的RADIUS accounting.

RADIUS accounting [RFC2866]

这一节探索RADIUS协议的accounting功能. accounting是一种方法来跟踪资源的使用和通常用于计费.

Operation

RADIUS accounting服务器运行在1813端口. 当一个用户的session开始, NAS发送一个Accounting-Request包给RADIUS服务器. 这个包必须包含某些AVPs. 他是在成功authentication之后发送的第一个包. 服务器端将会确定接收通过发送一个匹配的Accounting-Response包.

在整个session, NAS可以发送可选的update报告关于一个指定用户的时间和数据使用. 当用户的session结束, NAS通知服务器. 这个将关闭在用户的session期间的accounting细节.

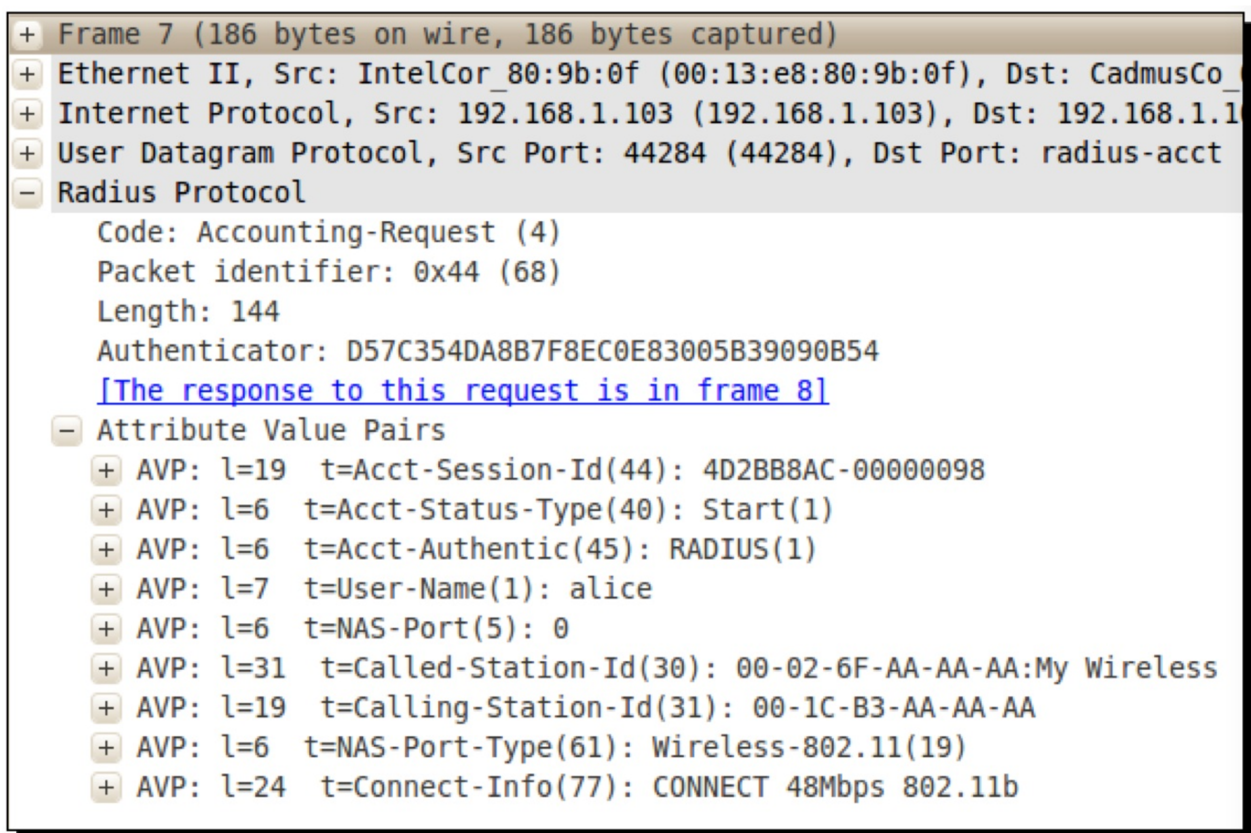
RADIUS客户端的功能为未来做准备比如服务器下线. NAS将会然后依赖他是如何配置的, 重试或者联系另一个RADIUS服务器.

当一个RADIUS服务器作为一个转发代理到另一个RADIUS服务器时, 他将会作为accounting数据的中继. 他也可能在转发前本地记录accounting数据.

包格式

Accounting包括RADIUS code 4(Accounting-Request)和code 5(Accounting-Response)包. Accounting包像authentication包一样使用相同的RADIUS协议. 一个独特的accounting包的特性是User-Password attribute不在请求中发送.

看下面的Wireshark输出, 显示一个典型的accounting 交易. 他从一个来自客户端的Accounting-Request开始.



服务器然后回复给客户端一个Accounting-Response.

```
+ Frame 8 (62 bytes on wire, 62 bytes captured)
+ Ethernet II, Src: CadmusCo_63:c2:83 (08:00:27:63:c2:
+ Internet Protocol, Src: 192.168.1.106 (192.168.1.106
+ User Datagram Protocol, Src Port: radius-acct (1813)
- Radius Protocol
    Code: Accounting-Response (5)
    Packet identifier: 0x44 (68)
    Length: 20
    Authenticator: A72D9494B5D5E987B9577C20AA5D965A
    [This is a response to a request in frame 7]
    [Time from request: 0.001801000 seconds]
```

Accounting-Request包也需要包含某些AVPs. 让我们看一下在accounting中使用的重要的AVPs.

Acct-Status-Type [Type40]

这个包表示用户或者NAS的状态. 一个NAS可能发送临时的更新关于某个session的使用. 为了做这个, NAS设置类型为Interim-Update. 这个允许我们来近似实时地跟踪使用趋势.

RADIUS服务器不会在一个NAS上检查. 如果一个NAS已经通知RADIUS服务器关于一个新连接的用户(status类型Start)并且然后NAS完全关闭, RADIUS服务器上的记录将会仍然指示那个用户连接在NAS上, 而事实上没有. 这些记录被称为欺骗条目(rogue entry). 为了减少欺骗条目, 最好让NAS发送一个Accounting-Off紧接着一个Accounting-On包, 仅仅在boot-up之后, 并且也发送一个Accounting-Off包在关闭之前. 这个动作将会导致RADIUS关闭所有打开的记录对于任何连接到指定NAS的用户允许一个干净的启动.

欺骗条目尤其称为问题当你限制一个用户可以拥有的session的时候. 如果限制一个用户session的组件利用包含欺骗条目的数据, 那么计算将会不准确.

下面的数字到类型表格可以被用来索引可能的状态值.

Decimal value	Status type
1	Start
2	Stop
3	Interim-Update
7	Accounting-On
8	Accounting-Off
9-14	Reserved for tunnel accounting
15	Reserved for failed

尽管Acct-Status-Type AVP不是必须的，他总是被包含.

Acct-Input-Octets [Type42]

这个表示在session期间接收到的byte并且带有Acct-Status-Type有一个值Interim-Update或者Stop.

注意值的限制. 4个byte限制他到4294967296. 大多数现代RADIUS实现已经满足这个需要.

Acct-Output-Octets [Type43]

这个表示在session期间发送的byte，并且带有Acct-Status-Type，值为Interim-Update或者Stop.

注意值的限制. 4个byte限制他为4294967296. 大多数现代RADIUS实现已经满足这个需要.

Acct-Session-Id [Type44]

对于所有的Accounting-Request包的规定动作，这个是一个唯一的值用来匹配Start, Interim 和Stop记录. 所有的一个session的Start, Interm, 和Stop记录应该有对于Acct-Session-Id相同的值.

Acct-Session-Time [Type46]

这个名字是自解释的. 以秒为时间表示session的长度，并且带有Acct-Status-Type，值为Interim-Update 或者Stop.

Acct-Terminate-Cause [Type49]

这个伴随着一个Acct-Status-Type AVP, 他的值设置为Stop. 这个AVP的值为十进制数. 他与Acct-Status-Type的行为类似, 用一个指定的数值对应解析一个终止原因.

Conclusion

这个带给我们RADIUS accounting的结束. 下一节我们将看某些RFC, 添加功能和增强给RFC2865和RFC2866的RADIUS定义.

RADIUS扩展

在初始的RFC定义了通常的RADIUS和RADIUS accounting, 许多扩展被建议来扩展RADIUS的使用或者提高一些弱点.

也有一个改进的RADIUS协议叫做Diameter(一个文字游戏--两倍好于RADIUS). 然后Diameter的接受非常缓慢, RADIUS仍然保持事实的标准对于可预见的未来. 一个主要的原因可能是事实上许多Diameter的增强已经被许多RADIUS的扩展实现. 例如, RadSec协议通过TCP和TLS传输RADIUS协议. 这个让RADIUS再漫游环境更容易扩展.

尽管有许多, 我们将只研究两个可能被使用的扩展.

Dynamic Authorization extension [RFC5176]

这个扩展帮助创建从RADIUS服务器到NAS的反馈回环. 这个实际上交换了客户端和服务器端之间的角色. RADIUS服务器成为NAS的一个客户端.

动态authorization允许RADIUS服务器来通知NAS关于已经做的改变给在NAS上一个用户的现有session. 有2个这种扩展的流行应用.

Disconnect-Message [DM]

也叫做一个POD(Packet Of Disconnect), 这个用来断开一个现有用户的session. RADIUS服务器发送断开连接请求, 并且NAS必须回复无论断开是否成功.

Change-of-Authorizaion Message [CoA]

这个消息提供数据来改变一个先有用户session的authorization. 例如, 我们现在可以动态改变每个session的带宽限制. 这让我们可以当互联网连接下降时可以增加每个session的带宽. 反之, 也可以.

MikroTik RouterOS包括这个功能在一些使用RADIUS的服务上.

下表列出包含的RADIUS包的code和名称

RADIUS code(decimal)	Packet type	Sent by
40	Disconnect-Request	RADIUS server
41	Disconnect-ACK	NAS
42	Disconnect-NAK	NAS
43	CoA-Request	RADIUS server
44	CoA-ACK	NAS
45	CoA-NAK	NAS

RADIUS support for EAP [RFC3579]

EAP代表Extensible Authentication Protocol. 他大多用于以太网交换机和Wi-Fi Access Point的安全.

EAP支持一个外部authorization服务器的使用. RADIUS可以是这样的一个服务器. EAP将会使用RADIUS协议来wrap EAP数据在AVP内部为了authenticate一个连接.

这本书有个专门的章节关于EAP, 由于他是Wi-Fi企业安全的如此一个重要的部分.

在下一节, 将会研究FreeRADIUS项目. FreeRADIUS是一种RADIUS协议的实现, 并且他的许多扩展包括2个在这里提到的.

FreeRADIUS是一个开源项目提供一个非常功能丰富的RADIUS协议的实现带有他的许多增强. 当人们提到FreeRADIUS的时候, 他们通常讨论服务器软件.

历史

FreeRADIUS的开发在1999年开始, 在原始的Livingston RADIUS服务器的未来变得不确定的时候. 这需要创建一个新的RADIUS服务器, 开源并且包含活跃的社区.

FreeRADIUS成功的获得一个可靠的名声, 并且能够打败大多数商业竞争者. 座右铭"世界上最流行的RADIUS服务器"已经没有可挑战德尔了, 使得他成为一个有效的陈述.

特色

FreeRADIUS有许多特色, 促进了他的流行. 让我们看一下:

- 开源: 不仅仅是免费. 你可以自由的适配, 修改, 扩展和需要的修复. FreeRADIUS在GPL协议下发布.
- 模块化: FreeRADIUS包含许多模块. 你可以创建的模块给FreeRADIUS使用. 模块包括LDAP集成或者SQL后端支持. 也有Perl和Python模块, 让你在FreeRADIUS上使用这两种强大的脚本语言.
- 被大量使用: 一个人不会因为选择了FreeRADIUS而被解雇. 很容易从ISP和在部署的FreeRADIUS上有大量用户的大公司得到帮助. FreeRADIUS执行了一个调查来决定FreeRADIUS的使用和部署大小. 调查的详细结果可以从他们的请求中得到.
- 活跃的社区: 因为FreeRaDIUS有这样大的用户基础, 很大的机会有其他人跟你遇到同样的问题. FreeRADIUS有活跃的邮件列表带有可搜索的文档.
- 可得到的信息: 信息可能不在一个地方, 但是可以得到, 只是需要被发现. 有许多的wiki包含很多细节. 也有许多man page和配置文件, 写的很详细和容易follow.
- 活跃地开发: FreeRADIUS遵循"release early, release often"的座右铭. RADIUS协议的新进展通常最先在FreeRADIUS中支持. 你可以期待每年有一个或者更多的FreeRADIUS release.
- 商业支持: FreeRADIUS的核心成员提供商业支持. 在FreeRADIUS有许多知识丰富的人可以提供付费支持. Network RADIUS SARL是一个好的网站带有更多的关于付费支持的细节: http://networkradius.com
- 可用性: FreeRADIUS可以被许多操作系统使用. 所有的流行的linux发行版包括他作为他们的可得到的包. 他甚至支持windows. FreeRADIUS网站上有二进制下载链接.

缺点

没有软件是完美的. FreeRADIUS也不例外. 这有一些他的缺点:

- 复杂: 这是唯一真的缺点. FreeRADIUS提供一个包含一切的软件, 带有许多配置选项. 如果你不小心, 你的系统会出问题.
- 漏洞: 过去一些漏洞被报道, 但是他们从那时已经被修复. 你可以了解每个FreeRADIUS的漏洞, 从他们的官网.

竞争者

当FreeRADIUS陈述他是最流行的服务器, 他和谁是竞争者呢? 有许多竞争RADIUS服务器和竞争技术. 竞争服务器包括Cisco's ACS, Microsoft's IAS, 和Radiator. 竞争AAA技术包括Diameter, TACACS+(CISCO专有, 尽管也被其他的企业网络设备支持), 和LDAP(LDAP只支持authentication).

这一章是引言和我们要将的基础. 作为讨论的重点, 确定知道下面的事实.

名称	代表	简称
AAA	Authentication, Authorization, Accounting	对于访问和使用的合适控制的必要的三个组件
NAS	Network Access Server	例如一个控制网络访问设备, 一个VPN服务器, 作为一个RADIUS客户端.
AVP	Attribute Value Pair	在RADIUS包中一个3域的组件来包含一个特殊的域和他的数据
VSA	Vendor-Specific Attributes	由指定vendor管理的扩展的AVP

- AAA是一个安全架构模型.
- RADIUS是AAA的一种特定实现.
- FreeRADIUS是RADIUS的实用应用.
- 因此我们有AAA -> RADIUS -> FreeRADIUS
- RADIUS是关于中心控制和NAS vendor支持的事实标准.
- RADIUS是一个 C/S 协议. 他使用UDP, 监听在1812端口用于authentication, 在1813端口用于accounting请求.
- RADIUS数据包有code域, 指定RADIUS包的类型.
- RADIUS数据包有0或者更多的AVPs, 包含用在RADIUS中的数据.
- FreeRADIUS实现了RADIUS协议, 还有在RFC中的许多扩展.
- FreeRADIUS是一个非常流行的, 广泛使用的, 非常灵活的RADIUS服务器.

这一章是FreeRADIUS的开始. 主要课程从下一章开始, 我们将安装和开始使用FreeRADIUS.

课堂作业 - RADIUS知识

1. 解释NAS device术语.
2. 什么是一个session的起点和终点?
3. RADIUS使用哪种协议和端口?
4. RADIUS客户端和服务器端需要成功的通信?
5. 当authenticate一个用户时, RADIUS客户端会发送什么?
6. 谁初始化一个Disconnect Request包并且谁接受他?
7. 命名AVP的三个组件?
8. Alice使用用户名 `alice@freeradius.org` 连接到一个网络. Alice术语的realm的名称是什么?

有两种方法安装FreeRADIUS到一个Linux服务器. 你已经简单低安装预编译的包或者从源码编译安装. 这一章会教你这两种方法.

在这一章我们将会:

- 从预编译包安装FreeRADIUS.
- 从源码编译和安装FreeRADIUS.
- 调查FreeRADIUS安装了哪些程序.
- 确保FreeRADIUS正确地运行.

有许多linux发行版可以选择. 我们将会选择三种流行的发行版来尽可能的包括所有听众来避免发行版战争.

基于他们使用系统的包管理系统, 许多linux发行版分成2组. 一组使用RPM(Red Hat Package Manager), 而另一组使用dpkg包管理器. 我们选择两种基于RPM的发行版, CentOS和SUSE, 在企业中流行. 取代使用Debian作为一个基于dpkg的发行版, 我们选择Ubuntu因为他在初学者中广泛流行. 由于Ubuntu派生于Debian, 讨论Ubuntu的章节也适用于Debian不需要大的修改.

这个章节的后面步骤需要安装

Distribution	Version
CentOS	5.5
SUSE	SLES 11
Ubuntu	10.4

一个典型的可以root访问的服务器是一个基础. 使用这个章节作为一个向导, 如果你有一个带有不同于那个指定版本的发行版.

如今的Linux发行版有许多预编译软件, 可以容易地安装. 一个单一的命令可以使用来从一个软件仓库来安装FreeRADIUS. 这个会解决依赖和安装所有需要的包为了展示一个工作的系统.

关于软件包管理系统: http://en.wikipedia.org/wiki/Package_management_system

3种发行版的默认安装将会包含软件仓库包含FreeRADIUS包.

预编译的FreeRADIUS包可以通过下面的命令在相应发行版进行安装.

- CentOS: `yum install freeradius2 freeradius2-utils`
- SUSE: `zypper in freeradius-server freeradius-server-utils freeradius-server-doc`
- Ubuntu: `sudo apt-get install freeradius`

优点

- 自动解决依赖. 这个包括考虑到未来的安全更新, 跟踪所有可选的包, 确保正确的版本依赖.
- linux distributor的QA测试确保软件正常工作.
- 更新会被linux distributor照顾好.
- 发行版相关的tweak已经执行.

使用预编译包的一个妥协是你不会用到最新的FreeRADIUS包到你的机器上.

额外的包

FreeRADIUS是一个功能丰富的软件. 不同的发行版展示他的FreeRADIUS不同地, 通过分散到多个不同的包里.

CentOS和Ubuntu包含某些FreeRADIUS服务器的包作为可选包. 这使得基本的服务器安装包保持瘦的. 安装可选的服务器模块包将会也安装需要的依赖. 这表示, 例如, 当你安装freeradius-mysql包的时候, 所有需要的MySQL库将作为依赖安装.

SUSE通过功能划分他们的包. 你可以发现客户端和服务器端都有他各自的包集合. SUSE也有FreeRADIUS的utilities和文档的包.

可得到的包

这一节列出每种发行版的可得到的预编译的FreeRADIUS包. 加粗的名字是推荐作为基本的FreeRADIUS安装的.

CentOS

包名	简短描述
freeradius2	高度可配置的RADIUS服务器
freeradius2-krb5	FreeRADIUS的Kerberos 5支持
freeradius2-ldap	FreeRADIUS的LDAP支持
freeradius2-mysql	FreeRADIUS的MySQL支持
freeradius2-perl	FreeRADIUS的Perl支持
freeradius2-postgresql	FreeRADIUS的PostgreSQL支持
freeradius2-python	FreeRADIUS的python支持
freeradius2-unixODBC	FreeRADIUS的Unix ODBC支持
freeradius2-utils	FreeRADIUS的utilities

SUSE

包名	简短描述
freeradius-client	FreeRADIUS的客户端软件
freeradius-client-libs	FreeRADIUS客户端的共享库
freeradius-server	Highly configurable RADIUS server
freeradius-server-dialupadmin	Web management for FreeRADIUS
freeradius-server-doc	FreeRADIUS的文档
freeradius-server-libs	FreeRADIUS的共享库
freeradius-server-utils	FreeRADIUS的客户端

注意, SUSE提供的 `freeradius-client` 包是被软件开发者使用来利用RADIUS来AAA. 客户端程序像 `radtest` 是包含在 `freeradius-server-utils` 包中.

Ubuntu

包名 | 简短描述 **freeradius** | FreeRADIUS的服务器端包 freeradius-dbg | 包含FreeRADIUS包的分离的调试符号 **libfreeradius2** | FreeRADIUS的共享库 freeradius-ldap | FreeRADIUS服务器的LDAP模块 **freeradius-common** | FreeRADIUS的普通文件, 包括字典和man pages freeradius-iodbc | FreeRADIUS服务器的iODBC模块 freeradius-krb5 | FreeRADIUS服务器的Kerberos模块 **freeradius-utils** | FreeRADIUS的客户端程序, 包括程序像radclient, radtest, smbencrypt, radsniff, 和 radzap freeradius-postgresql | FreeRADIUS服务器的PostgreSQL模块 freeradius-mysql | FreeRADIUS服务器的MySQL模块 freeradius-dialupadin | web管理插件 libfreeradius-dev | FreeRADIUS的共享开发库

特殊的考虑

旧版本的Ubuntu的预编译包没有编译进去SSL库支持. 当安装FreeRADIUS到这些版本的时候, 如果你需要在一些EAP扩展上支持SSL, 你需要自己编译.

SUSE也提供 `yast -i` 命令来安装软件. 但使用 `zypper` 替代, 因为当安装软件和依赖时, 他有更好的决策能力.

在CentOS中FreeRADIUS包的名称是 `freeradius2` 而不是 `freeradius` . 这是因为最开始在CentOS 5上支持的FreeRADIUS版本是1.1.3. FreeRADIUS的1.x和2.x之间的配置文件有改变, 所以需要在名称上有所改变.

不是所有的FreeRADIUS模块有他们在Ubuntu或CentOS上匹配的包. 没有匹配的包的模块就简单低包含在FreeRADIUS的主包上.

留意防火墙

CentOS和SUSE默认安装一个活跃的防火墙. 请确保UDP端口1812和1813是对外开放的.

CentOS

有一个程序来配置CentOS上的防火墙, 叫做 `system-config-securitylevel-tui` , 需要root来运行. 这个会启动一个基于鼠标的程序. 选择 `Customize` 选项然后按下 `Enter` . `Allow incoming | Other Ports` 列表应该包括 `1812:udp 1813:udp` . 选择 `OK` 来返回到主界面, 然后再选择 `OK` , 来提交修改.

确认端口是开放的通过下面命令的输出:

```
# iptables -L -n | grep 181*
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           state NEW udp dpt:1812
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           state NEW udp dpt:1813
```

不推荐你用下面的命令来关闭防火墙.

```
# /etc/init.d/iptables save
# /etc/init.d/iptables stop
# /sbin/chkconfig iptables off
```

为了确认是否防火墙是关闭的, 使用下面的命令.

```
# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

SUSE

在SUSE上配置防火墙可能有点a catch-22 situation由于默认的防火墙是如此安全, 你甚至不能用SSH进入系统! 进入SLES服务器然后开始 `YaST` . 选择 `Security and Users | Firewall` . 选择左边的 `Allowed Services` . 我建议你添加 `Secure Shell Server` 到 `External Zone` . 点击 `Advanced` 按钮, 然后添加1812, 1813到 `UDP Ports` . 点击 `OK` . 点击 `Next` 和 `Finish` 来提交这些修改.

通过下面的命令来确认端口是开放的.

```
# iptables -L -n | grep 181
ACCEPT     udp  --  0.0.0.0/0        0.0.0.0/0          udp dpt:1812
ACCEPT     udp  --  0.0.0.0/0        0.0.0.0/0          udp dpt:1813
```

不推荐用下面的方法关闭防火墙.

1. 使用 `YaST` 然后选择 `Security and Users | Firewall` .
2. 选择左边的 `Start-Up` , 选择右边的 `Disable Firewall Automatic Starting` . 也选择 `Stop Firewall Now` , 然后按 `Enter` 键来停止当前运行的防火墙.
3. 点击 `Next` 和 `Finish` 来提交修改.

通过下面的命令来确认防火墙现在是关闭的.

```
# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

从源码安装软件是 `configure, make, make install` 的同义词. 我们将使用发型吧的包管理器来从源码编译新的软件.

下一节是可选的内容如果你已经安装了FreeRADIUS的预编译的包.

从源码编译

有时有需要来安装最新版本的软件或者包含一些预编译软件没有支持的模块. 这需要从源码编译软件. 大多数开源软件包是作为一个TAR压缩包发布的. TAR实际上可以指用于创建TAR文件的程序(名称是tape archive的简写)或者合成文件的文件格式. 一个TAR文件也通常是压缩的来减少体积. 文件名将会表示压缩的格式. 一个文件名以 `.tgz` 或者 `.tar.gz` 结尾使用 `gzip` . 一个文件名以 `.tbz` , `.tb2` , 或者 `.tar.bz2` 结尾使用 `bzip2` 压缩. 为了编译软件我们解压这个文件, 然后执行命令 `configure` , `make` 和 `make install` 在解压的文件夹内.

然而有更好的方式利用发行版的包管理系统来编译软件. 不是所有的TAR文件帮助这个容易实现, 但是FreeRADIUS是这样一个成熟的项目, 允许在任意这3个发行版上从源码编译软件.

编译包的优势

下面是一些从源码编译包的优势

- 容易安装, 升级, 卸载或发布软件.
- 容易检查那个版本的软件是安装的.
- 能够看到哪些文件被安装, 哪些发行版相关的修改.

既然你已经确定用这种方式编译包, 我们将会为每个发行版来做他.

创建编译环境

不用生产机器来开发是一个好的实践. 这包括从源码编译软件. 创建一个你可以编译包的虚拟机. 测试之后简单低拷贝和安装新的包到一个生产机器.

CentOS

CentOS是一个RedHat风格的发行版. 他是RHEL的社区版. CentOS不是有RedHat直接赞助像Fedora项目那样. 尽管RedHat的品牌和logo从CentOS移除, 但是它跟RHEL是一样的, 由于相同的源代码用来生成发行版.

CentOS使用RPM包管理器来管理软件. 安装的软件包叫做RPM.

编译CentOS RPMs

这一节使用从FreeRADIUS的wiki页面的推荐和指令来创建最新的RPM到CentOS上. http://wiki.freeradius.org/Red_Hat_FAQ#How_to_build_an_SRPM

你必须用root用户来执行大多数命令. 一旦完成, 遵循这些步骤.

1. 安装 `rpm-build` 包, 这个创建的编译包需要的目录结构. `yun install rpm-build`
2. 允许普通用户对这个目录有写和执行权限. `find /usr/src/redhat -type d | xargs chmod a+wx`
3. Fedora上可以得到FreeRADIUS的最新版本. 我们可以使用这个版本来在CentOS上编译RPM. 从下面的URL上下载RPM源码. https://kojipkgs.fedoraproject.org//packages/freeradius/2.2.5/2.fc19/src/freeradius-2.2.5-2.fc19.src.rpm RPM的源码包以 `.src.rpm` 结尾.
4. 安装这个源码包. 例如使用下面的命令. 你的源码包的命名可能不同. `rpm -ihv --nomd5 freeradius-2.2.5-2.fc19.src.rpm` .
5. 使用下面的命令来决定需要的依赖. 这些依赖必须在包边以前被安装. `rpmbuild -ba /usr/src/redhat/SPECS/freeradius.spec`
6. 使用 `yum` 来安装这些依赖. 在我的系统上时下面这些. `yum install autoconf gdbm-devel libtool libtool-ltdl-devel openssl-devel pam-devel zlib-devel net-snmp-devel net-snmp-utils readline-devel libpcap-devel openldap-devel krb5-devel python-devel mysql-devel postgresql-devel unixODBC-devel`
7. 安装一个编译器. `yum install gcc`
8. 再次运行 `rpmbuild` 命令. 使用普通用户来做, 使用 `tee` 来输出到一个文件来为了将来的研究. 这将会启动编译过程, 需要花费一些时间. `rpmbuild -ba /usr/src/redhat/SPECS/freeradius.spec | tee /tmp/freeradius_build_out.txt` .
9. 当它成功完成, 将会有一个指示RPM已经写入. 新安装的RPM将会得到下面这行. `Wrote: /usr/src/redhat/RPMS/i386/freeradius-debuginfo-2.1.10-1.i386.rpm`
10. 安装新编译的RPM. 下面的命令将会导致一个基本的FreeRADIUS安装. `yum --nogpgcheck install /usr/src/redhat/RPMS/i386/freeradius-2.1.10-1.i386.rpm /usr/src/redhat/RPMS/i386/freeradius-utils-2.1.10-1.i386.rpm` 如果你得到消息通知你 `No package <path and package name> available` , 确认你指定了正确的版本号.

发生了什么?

我们刚刚创建和安装了CentOS上最新的RPM. 下面是一些重点的分解.

安装 `rpm-build`

`rpm-build` 包包含脚本和可执行程序用来编译RPM包. 他包括 `rpmbuild` 程序, 我们将用来创建RPM.

RPM包的源码.

我们使用FreeRADIUS wiki上推荐的Fedora的RPM包源码. 这导致一个更加稳定和更好的包工作集合, 已经包含了发行版相关的tweak和修改.

包名

注释包名是 `freeradius` 而不是 `freeradius2` . 这是因为所有的在Fedora里面的包已经是2.x.

从FreeRADIUS 1.x版本升级在Fedora系统不像CentOS那样没有问题.

更新一个已经存在的安装

如果你已经有FreeRADIUS安装了, 他可能包含一些模块你需要包含在更新的时候.

- 决定FreeRADIUS的版本和当前安装的包通过使用下面的命令. `yun list freeradius\*`
- 当你已经作为预编译包安装了 `freeradius2` , `yum` 将会抱怨, 当尝试安装新的 `freeradius` 包, 因为名字冲突. 你可以卸载 `freeradius2` 包, 通过下面的命令. `yum erase freeradius\*`
- 上面假定你没有做任何配置修改到安装, 你想要保留.

CentOS

CentOS是一个RedHat风格的发行版. 他是RHEL的社区版. CentOS不是有RedHat直接赞助像Fedora项目那样. 尽管RedHat的品牌和logo从CentOS移除, 但是它跟RHEL是一样的, 由于相同的源代码用来生成发行版.

CentOS使用RPM包管理器来管理软件. 安装的软件包叫做RPM.

编译CentOS RPMs

这一节使用从FreeRADIUS的wiki页面的推荐和指令来创建最新的RPM到CentOS上. http://wiki.freeradius.org/Red_Hat_FAQ#How_to_build_an_SRPM

你必须用root用户来执行大多数命令. 一旦完成, 遵循这些步骤.

1. 安装 `rpm-build` 包, 这个创建的编译包需要的目录结构. `yun install rpm-build`
2. 允许普通用户对这个目录有写和执行权限. `find /usr/src/redhat -type d | xargs chmod a+wx`
3. Fedora上可以得到FreeRADIUS的最新版本. 我们可以使用这个版本来在CentOS上编译RPM. 从下面的URL上下载RPM源码. https://kojipkgs.fedoraproject.org//packages/freeradius/2.2.5/2.fc19/src/freeradius-2.2.5-2.fc19.src.rpm RPM的源码包以 `.src.rpm` 结尾.
4. 安装这个源码包. 例如使用下面的命令. 你的源码包的命名可能不同. `rpm -ihv --nomd5 freeradius-2.2.5-2.fc19.src.rpm` .
5. 使用下面的命令来决定需要的依赖. 这些依赖必须在包边以前被安装. `rpmbuild -ba /usr/src/redhat/SPECS/freeradius.spec`
6. 使用 `yum` 来安装这些依赖. 在我的系统上时下面这些. `yum install autoconf gdbm-devel libtool libtool-ltdl-devel openssl-devel pam-devel zlib-devel net-snmp-devel net-snmp-utils readline-devel libpcap-devel openldap-devel krb5-devel python-devel mysql-devel postgresql-devel unixODBC-devel`
7. 安装一个编译器. `yum install gcc`
8. 再次运行 `rpmbuild` 命令. 使用普通用户来做, 使用 `tee` 来输出到一个文件来为了将来的研究. 这将会启动编译过程, 需要花费一些时间. `rpmbuild -ba /usr/src/redhat/SPECS/freeradius.spec | tee /tmp/freeradius_build_out.txt` .
9. 当它成功完成, 将会有一个指示RPM已经写入. 新安装的RPM将会得到下面这行. `Wrote: /usr/src/redhat/RPMS/i386/freeradius-debuginfo-2.1.10-1.i386.rpm`
10. 安装新编译的RPM. 下面的命令将会导致一个基本的FreeRADIUS安装. `yum --nogpgcheck install /usr/src/redhat/RPMS/i386/freeradius-2.1.10-1.i386.rpm /usr/src/redhat/RPMS/i386/freeradius-utils-2.1.10-1.i386.rpm` 如果你得到消息通知

你 `No package <path and package name> available` , 确认你指定了正确的版本号.

发生了什么?

我们刚刚创建和安装了CentOS上最新的RPM. 下面是一些重点的分解.

安装 `rpm-build`

`rpm-build` 包包含脚本和可执行程序用来编译RPM包. 他包括 `rpmbuild` 程序, 我们将用来创建RPM.

RPM包的源码.

我们使用FreeRADIUS wiki上推荐的Fedora的RPM包源码. 这导致一个更加稳定和更好的包工作集合, 已经包含了发行版相关的tweak和修改.

包名

注释包名是 `freeradius` 而不是 `freeradius2` . 这是因为所有的在Fedora里面的包已经是2.x.

从FreeRADIUS 1.x版本升级在Fedora系统不像CentOS那样没有问题.

更新一个已经存在的安装

如果你已经有FreeRADIUS安装了, 他可能包含一些模块你需要包含在更新的时候.

- 决定FreeRADIUS的版本和当前安装的包通过使用下面的命令. `yun list freeradius\*`
- 当你已经作为预编译包安装了 `freeradius2` , `yum` 将会抱怨, 当尝试安装新的 `freeradius` 包, 因为名字冲突. 你可以卸载 `freeradius2` 包, 通过下面的命令. `yum erase freeradius\*`
- 上面假定你没有做任何配置修改到安装, 你想要保留.

SUSE也利用RPM包管理器来软件管理就像RedHat发行版一样. 尽管都使用RPM作为基础的软件管理, SLES使用 `zypper` 来管理软件包仓库, CentOS使用 `yum` .

SLES是一个硬核的企业发行版. 尽可能使用预编译包, 但是如果你感到勇敢, 想要你的手变脏, 这一节为你准备.

在SLES上编译FreeRADIUS的RPM包, 要比CentOS或者Ubuntu徐璈更多的努力.

添加一个OpenSUSE仓库

从源码编译FreeRADIUS需要许多开发库. 标准的SLES仓库不需要全部包含他们他们, 甚至当使用的SDK DVD是SLES的一部分. 我们因此需要添加OpenSUSE仓库. 这将会满足所有的开发库需求.

1. 通过下面的URL定位离你最近的OpenSUSE镜像: http://mirrors.opensuse.org
2. 使用root用户开始 `YAST` .
3. 选择 `Software | Software Repositories` 并且然后选择 `Add` 来添加到一个仓库.
4. 添加在开始时你选择的OpenSUSE镜像. 对于我的OpenSUSE11.3最近的URL是: ftp://opensuse.mirror.ac.za/opensuse/distribution/11.3/repo/oss/suse
5. 你可能要被接受GnuPG密钥的导入对于新的仓库. 选择 `Import` 来继续.

这个设置编译FreeRADIUS RPM的阶段. 现在遵循下面的步骤:

1. 从官网下载最新的 `.tar.bz2` 格式的FreeRADIUS源码.
2. SLES包括一个目录结构专门用于编译RPM. 位于 `/usr/src/packages` . 拷贝原始的 `bz2` 源文件到SOURCES目录. 替换x和y为对应的版本. `cp freeradius-server-2.x.y.tar.bz2 /usr/src/packages/SOURCES/` .
3. 从TAR文件解压SUSE RPM .spec文件, 替换x,y为对应版本.

```
# cd /usr/src/packages/SOURCES/
# tar -xvjf freeradius-server-2.x.y.tar.bz2 freeradius-server-2.x.y/suse/freeradius.s
```

4. 拷贝下面的文件到 `/usr/src/packages/SPECS` 目录:

```
# cp freeradius-server-2.x.y/suse/freeradius.spec ../SPECS/
```

5. 编辑 `spec` 文件的下面的行, 修改 `%{fillup_and_insserv -s freeradius START_RADIUSD}` 为 `%{fillup_and_insserv freeradius}` .
6. 运行下面的命令来决定依赖. `rpmbuild -ba /usr/src/packages/SPECS/freeradius.spec`
7. 这个将会列出必要的开发包来安装, 为了FreeRADIUS RPM被编译. 在你的系统上的列表可能不同. 下面是我的系统上的结果(用 `zypper` 安装他们)

```
zypper in db-devel e2fsprogs-devel gcc-c++ gdbm-devel gettext-devel glibc-devel ncurs
```

8. 再次运行 `rpmbuild` 命令. 如果所有的依赖都满足, RPM的编译将会开始. 把结果tee到一个文件用于将来的调查.

```
rpmbuild -ba /usr/src/packages/SPECS/freeradius.spec | tee /tmp/build_out.txt
```

9. 当编译完成, RPM位于 /usr/src/packages/RPMS/<architecture>/ 目录.
10. 使用下面命令安装新的FreeRADIUS包.

```
# cd /usr/src/packages/RPMS/<architecture>/
# zypper in freeradius-server-2.x.y-0.i586.rpm freeradius-server-libs-2.x.y-0.i586.rp
```

11. 注意默认FreeRADIUS将会已用户 `radiusd` 运行. 这个用户是在安装FreeRADIUS过程中创建的. 给这个用户 `certs` 目录的权限. 否则会安装失败.

```
chown -R radiusd. /etc/raddb/certs
```

12. 通过 `radiusd -X` 来确认 `radiused` 可以正确启动.
13. `Ctrl+C` 将会停止正在运行的FreeRADIUS. 你现在可以使用下面的启动脚本来启动FreeRADIUS

```
/etc/init.d/freeradius start
```

发生了什么?

我们完成了在SLES上编译和安装FreeRADIUS. 尽管不像CentOS那样优雅, 仍然没有许多问题.

zypper还是yast -i

使用 `zypper` 来安装需要的依赖. 使用 `yast -i` 来安装需要的 `gettext-devel` 包将会返回一个错误. `zypper` 知道 `gettext-tools` 包将会满足依赖.

使用 `yast -i` 来安装依赖也导致从OpenSUSE仓库不需要的更新. 你会被警告.

手动tweak

有一些tweak我们必须手动执行, 为了获取在SLES上最新的FreeRADIUS.

- 添加OpenSUSE仓库作为一个开发包源.
- 从OpenSUSE安装各种开发包.
- 编辑 `freeradius.spec` 来卸载旧的宏.
- 修改包含证书的目录的权限.

Ubuntu是基于Debian Linux, 使用dpkg包挂利器来管理软件像deb.

`apt` 程序是用来管理dpkg仓库, 跟 `zypper` 和 `yum` 使用RPM是一样的.

下面的步骤将会展示如何安装deb.

1. 安装 `dpkg-dev` 包. 这个包提供开发工具(包括 `dpkg-source`), 用来unpack, 编译, 和上传Debian源码包. `sudo apt-get install dpkg-dev` 如果这个包没有安装, 检查被 `apt` 程序使用的源列表. 这个仓库的源定义在 `/etc/apt/sources.list` 文件, 并且通过 `apt-get update` 命令更新.
2. 安装所有 `freeradius` 包需要的开发库

   ```
   sudo apt-get build-dep freeradius
   ```

   如果你得到一个信息: `Unable to find a source package for freeradius` , 可能因为仓库源(deb-src)没有包含在 `/etc/apt/sources.list` 文件.
3. 确保 `fakeroot` 程序被安装.

   ```
   sudo apt-get install fakeroot
   ```

4. 确保 `ssl-cert` 包被安装. 不能安装这个包将会导致编译EAP模块出现问题.

   ```
   sudo apt-get install ssl-cert
   ```

5. 下载最新版本的FreeRADIUS源码.

   ```
   wget ftp://ftp.freeradius.org/pub/freeradius/freeradius-server-2.x.y.tar.gz
   ```

6. 解压FreeRADIUS的源码TAR文件.

   ```
   tar -xzvf freeradius-server-2.x.y.tar.gz
   ```

7. 切换到解压目录.

   ```
   cd freeradius-server-2.x.y
   ```

8. 在当前目录创建一个 `fakeroot` 环境, 并且使用已经包含在 `debian` 子目录的控制文件来编译包. 编译过程将会花费一些时间并且依赖计算机能性能. 你可以监控这个命令的结果,是否有fatal错误, 尽管他应该能够工作.

   ```
   fakeroot dpkg-buildpackage -b -uc
   ```

9. 这将会创建所有需要的deb并且把他们放到 `freeradius-server-2-x.y` 目录外.
10. 通过下面的命令安装新的deb.

```
cd ../
sudo dpkg -i freeradius_2.x.y+git_i386.deb freeradius-common_2.x.y+git_all.deb freera
```

发生了什么?

我们刚刚创建和安装了最新的FreeRADIUS包到Ubuntu. 下面是一些分解的重点.

安装dpkg-dev

`dpkg-dev` 包安装编译deb的需要的所有工具. 这包括 `autoconfig` 和 `make` 工具, 还有一个编译器. 也包括 `dpkg-buildpackage` 陈天旭, 我们将会使用来创建deb.

使用build-dep

`apt-get` 是接下来使用的命令来指导他做什么. 当我们安装包时, 使用 `install` 命令, `build-dep` 命令用来安装所有一个指定源码包的依赖.

我们没有使用随着ubuntu一起的源码包, 而是使用来自FreeRADIUS最新的源码包. 这表示他不是一个十分安全的方法, 尤其当最新的源码包含开发库跟ubuntu默认装的版本不一致的时候. 然而你将会被通知在编译过程中, 可以最后安装这个依赖的库.

fakeroot

这个命令允许我们作为一个普通用户编译deb, 并且是推荐的方式. `Fakeroot` 跟着一个命令. 这个 `fakeroot` 调用的命令将会认为他是通过root用户有root权限对于文件操作. 在我们的例子中, 我们使用 `dpkg-buildpackage` 命令.

dpkg-buildpackage

这个命令在FreeRADIUS源码目录运行. 他将会利用 `debian` 目录的指定信息来编译源码和术后创建各种FreeRADIUS包. `-b` 选项是对于只有二进制的编译, 而 `-uc` 选项是跳过一个 `gpg` 密钥的签名.

安装debs

我们安装基础的包. 当我们需要额外的二模考时, 他们可以作为一个包来安装. 需要的依赖可以通过使用 `apt` 命令来安装.

对于那些守旧派的人

尽管我们鼓励这样编译包，我们没有阻止你使用 `configure, make, make install` 模式. 如果你运行 `./configure --help` 你会看到一个选项列表. 你可以用来tweak FreeRADIUS的编译过程. 你可以例如指定某个目录(--bindir=/usr/bin), 使能或禁用某个特性(--enable-developer), 或者包括某个包(--with-openssl)

FreeRADIUS有许多可执行文件被安装. 当在不同发行版之间移动时, 有小的不同点需要考虑.

一个这样的不同是配置文件的位置. 另一个不同是FreeRADIUS服务器程序的名称的不同. 在Ubuntu(和Debian)上, 他叫做 `freeradius` . 在CentOS和SLES, 他叫做 `radiusd` . 下表列出了重要的可执行程序, 带有一个简短描述.

名称	描述
/usr/sbin/raddebug	radmin的shell脚本wrapper用于调试输出不必在debug模式下运行 `radiusd` .
/usr/sbin/radiusd 或 /usr/sbin/freeradius	RADIUS服务器程序
/usr/sbin/radmin	连接到一个运行的 `radiusd` daemon的管理程序
/usr/bin/radclient	用来发送各种RADIUS包到RADIUS服务器和显示回复
/usr/bin/radconf2xml	以XML格式显示当然服务器的配置
/usr/bin/radcrypt	加密或者检查密码以DES或MD5格式
/usr/bin/radeapclient	发送EAP包给一个RADIUS服务器
/usr/bin/radlast	系统的上一个命令的前端用于显示从accounting 日志文件的输出
/usr/bin/radsqlrelay	用于管理记录在一个SQL日志文件的账户细节. 这个文件是 `rlm_sql_log` 模块创建的
/usr/bin/radtest	发送 `Access-Request` (code 1)包给一个RADIUS服务器, 并且显示回复. `radclient` 的前端
/usr/bin/radwho	从 `radutmp` 文件显示活跃的连接
/usr/bin/radzap	shell脚本wrapper用于移除在session数据库(file或者SQL)里的欺骗记录(rogue entries)
/usr/bin/smbencrypt	给定一个明文密码的 `nt` 密码哈希, 被MS-CHAP需要.

不是所有的列出的命令默认都可以执行. 命令像 `radwho` 和 `radlast` 依赖日志文件的存在, 因为他们作为这些命令的输入. 命令像 `radmin` 和 `raddebug` 需要一个在服务器上的特殊的设置在他们能够运行之前. 如果不是所有的这些命令都能在新的安装上运行, 不要担心.

FreeRADIUS应该被尽可能低的权限运行在生产环境中. 一个正常的安装创建一个专用的用户和组为了这个目的.

在CentOS和SLES上, 用户和组叫做 `radiusd` . 在Ubuntu上, 用户和组叫做 `freerad` . 只有特殊的配置才需要root权限.

在一些安装上(SLES), FreeRADIUS客户端程序不能读取字典文件. 这只有当你用一个普通用户运行时才会发生. 访问 `dictionary` 文件是需要的对于客户端来解析许多RADIUS AVPs到数字.

当这个问题存在时, 你将会的到下面的信息:

```
radclient: dict_init: Couldn't open dictionary "/etc/raddb/dictionary": Permission denied
```

通过使用下面命令来修复该问题.

```
# chmod o+xr /etc/raddb
# chomod o+r /etc/raddb/dictionary
```

在这之后一个普通用户应该可以运行FreeRADIUS客户端程序没有任何权限问题.

执行 `radiusd -X` (或者 `freeradius -X` 在Ubuntu), 用root用户, 将会启动FreeRADIUS以debug模式, 并且会给出提示, 如果有任何问题.

使用 `Ctrl + C` 来停止服务器程序.

如果你得到一个错误信息关于1812端口已经被使用, FreeRADIUS已经运行. 你可以通过启动脚本关闭然后再运行来解决.

我们也需要确保FreeRADIUS在重启后能够启动带有其余的服务. 不幸的是每个发行版在这里工作不同. 下表可以被用做一个向导来管理FreeRADIUS的启动.

这些脚本的每一个可以被列出的参数之一调用.

发行版	启动脚本
CentOS	`/etc/init.d/radiusd start/stop/restart`
SLES	`/etc/init.d/freeradius start/stop/restart`
Ubuntu	`/etc/init.d/freeradius start/stop/restart`

发行版	使能和禁用
CentOS	`/sbin/chkconfig radiusd on/off`
SLES	`/sbin/chkconfig -a freeradius` 和 `/sbin/chkconfig -d freeradius`
Ubuntu	`update-rc.d freeradius defaults` 和 `update-rc.d -f freeradius remove`

确保FreeRADIUS在运行: `pidof freeradius` 或者 `ps aux | grep radius`

检查FreeRADIUS在使用哪个网络接口和UDP端口: `netstat -unap | grep radius`

这一章是关于安装FreeRADIUS到你的机器.

- 安装预编译包. 最快和确定的方式来拥有一个工作的FreeRADIUS服务器.
- 从源码编译和安装FreeRADIUS. 这个会有点复杂, 但是使你可以运行最新的版本.
- FreeRADIUS安装各种可执行程序并且包含各种模块. 有些模块是单独打包的, 基于不同的linux发行版.
- 你也用过安装客户端程序. 让你高效低测试和排查问题.
- 用普通用户运行客户端程序导致文件权限问题, 当尝试访问字典问题. 这个可以通过修改权限来修复.

我们也讨论了如何确保重启后自动启动, 以debug模式运行FreeRADISU来排查问题.

既然FreeRADIUS安装了并且运行在我们的机器上, 我们可以开始使用它. 开始使用FreeRADIUS是下一章的主题.

课堂作业 - 安装

1. Isaac是困惑的. 他安装FreeRADIUS到Ubuntu使用跟CentOS相同的方式, 但是radiusd二进制程序不见了, 这是为什么?
2. 在你安装FreeRADIUS的预编译程序之后, 你重启服务器. 你想要运行 `radiusd` 以debug模式, 但是 `radiusd` 抱怨, 地址以及被占用, 这是什么问题?
3. 你想要FreeRADIUS来连接到一个CentOS上的MySQL数据库. 预编译的FreeRADIUS包需要什么?
4. 你的一个朋友请求帮助. 他想要在SLES11上编译FreeRADIUS, 到那时他尝试安装需要的编译库, 他不能找到所有的需要的库. 这是什么问题?

在FreeRADIUS被安装之后，他需要按照我们的需要进行配置. 这一章将会帮助你熟悉FreeRADIUS. 假定你已经了解第一章中讨论的RADIUS协议的基础了.

正在这一章我们将会:

- 对FreeRADIUS进行一个基本配置，并测试他.
- 发现更多获取帮助的方法.
- 学会推荐的配置和测试FreeRADIUS的方法.
- 了解一起是怎么适配在一起与FreeRADIUS工作的.

开始之前

这章假定你有一个FreeRADIUS的干净的安装. 你将会需要root权限来修改FreeRADIUS的配置文件来基本配置和测试

我们以一个下面简单的FreeRADIUS配置来开始这一章

- 本机作为一个NAS设备(RADIUS客户端)
- Alice定义为一个测试用户.

在我们已经定义客户端和测试用户之后，我们将会使用 `radtest` 程序来作为RADIUS客户端的角色并且测试Alice的authentication.

FreeRADIUS是通过修改配置文件来配置的. 这些文件的位置依赖于FreeRADIUS是怎样被安装的:

- 如果你安装发行版提供的标准FreeRADIUS包, 在CentOS和SLES上, 将会在 `/etc/raddb` . 在Ubuntu上将会在 `/etc/freeradius` .
- 如果你使用发行版的包管理系统从源码编译和安装FreeRADIUS, 在CentOS和SLES上, 将也会在 `/etc/raddb` . 在Ubuntu上将会在 `/etc/freeradius` .
- 如果你通过 `configure` , `make` , `make install` 来编译和安装FreeRADIUS, 他将会在 `/usr/local/etc/raddb` .

写明的命令假定FreeRADIUS的配置目录为你的当前工作目录.

1. 确保你是root用户为了能够编辑配置文件.
2. FreeRADIUS包括一个默认的客户端叫做 `localhost` . 这个客户端被在localhost上RADIUS客户端程序所使用来帮助排查问题和测试. 确认下面的入口存在在 `clients.conf` 文件中:

```
client localhost {
 ipaddr = 127.0.0.1
 secret = testing123
 require_message_authenticator = no
 nastype = other
}
```

3. 定义Alice作为一个FreeRADIUS测试用户. 添加下面的行到 `users` 文件的顶部. 确定第二行和第三行前面由一个单一的tab键缩进:

```
"alice" Cleartext-Password := "passme"
 Framed-IP-Address = 192.168.1.65,
 Reply-Message = "Hello, %{User-Name}"
```

4. 以debug模式启动FreeRADIUS. 使用启动脚本来关闭FreeRADIUS来确保没有其他实例在运行. 在Ubuntu中我们这样做:

```
# /etc/init.d/freeradius stop
# freeradius -X
```

   你也可以使用更暴力的方法: `kill -9 $(pidof freeradius)` 或者 `killall freeradius` 在Ubuntu上和 `kill -9 $(pidof radius)` 或者 `killall radiusd` 在CentOS和SLES上, 如果启动脚本没能停止FreeRADIUS.
5. 确保FreeRADIUS已经正确的启动通过确认在你的屏幕上的最后一行是: `Ready to process requests.` 如果这个没有发生, 阅读FreeRADIUS的debug输出.
6. 使用下面的命令来Authenticate Alice.

```
radtest alice passme 127.0.0.1 100 testing123
```

7. FreeRADIUS的debug输出将会展示 `Access-Request` 包到达并且FreeRADIUS服务器如何响应这个请求.
8. `radtest` 将也会显示FreeRADIUS服务器的回复:

```
Sending Access-Request of id 17 to 127.0.0.1 port 1812
 User-Name = "alice"
 User-Password = "passme"
 NAS-IP-Address = 127.0.0.1
 NAS-Port = 100
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=147, length=40
 Framed-IP-Address = 192.168.1.65
 Reply-Message = "Hello, alice"
```

发生了什么?

我们已经在FreeRADIUS上创建了一个测试用户. 我们也使用 `radtest` 命令作为一个客户端到FreeRADIUS服务器来测试authentication.

让我们详细描述一些有趣和重要的点.

配置FreeRADIUS

FreeRADIUS服务器的配置逻辑上划分为不同的文件. 这些文件被修改来配置某个功能, 组件或者FreeRADIUS模块. 然而有一个主配置文件包含各种子配置文件, 叫做 `radiusd.conf` .

默认的配置适合大多数安装. 只需要非常少修改来使FreeRADIUS在你的环境有用.

客户端

尽管在FreeRADIUS服务器的配置目录内有许多文件, 只有少量将来需要修改的. `clients.conf` 文件是用来定义到FreeRADIUS服务器的客户端的.

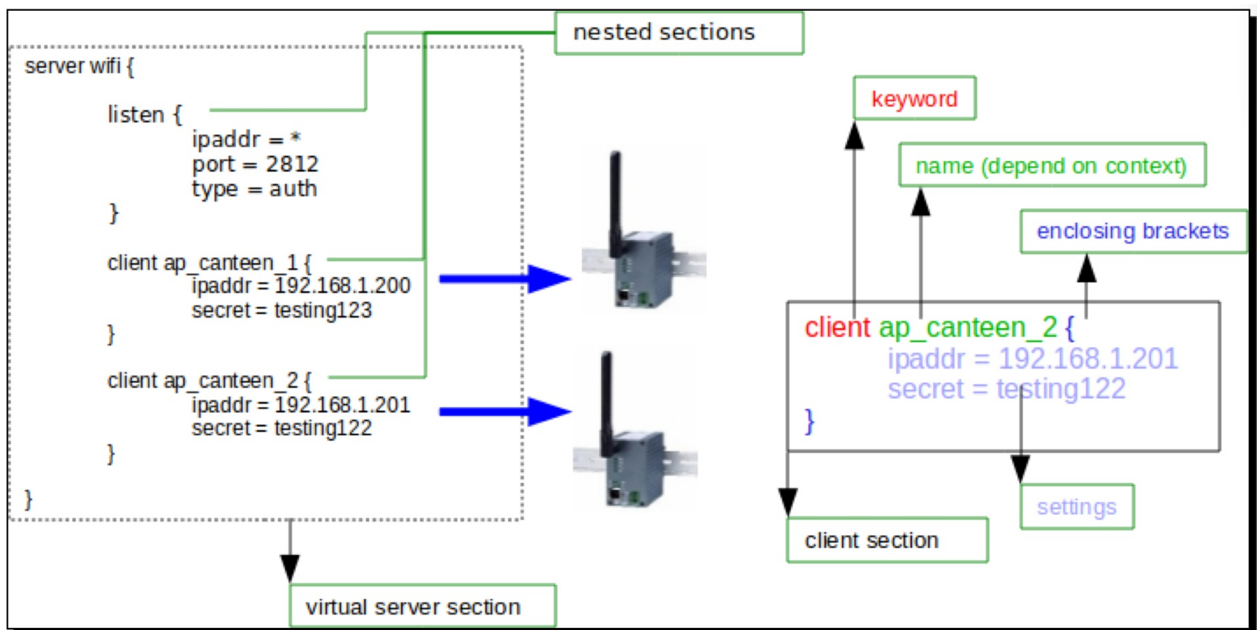
在一个NAS可以使用FreeRADIUS服务器之前, 他必须被定义为在FreeRADIUS服务器上的一个客户端. 让我们看一些关于客户端定义的要点.

Sections

一个客户端通过一个 `client` section定义. FreeRADIUS使用section来分组和定义许多东西. 一个section以一个关键字开始表示section名称. 然后跟着一个封闭的大括号. 在封闭的括号内部有许多配置指定到这个section. section可以被嵌套.

有时section的关键字跟着一个单一的词来区分通一个类型的不同的section. 这允许我们在 `clients.conf` 中有不同客户端入口. 没一个客户端有一个简短的名称来区分他和其他的.

`clients.conf` 文件不是唯一的文件, `client` section可以被定义, 尽管他是常用和最符合逻辑的地方. 下图显示在一个server section内嵌套定义的客户端.



客户端标识

FreeRADIUS服务器通过他的IP地址标识一个客户端. 如果一个不知道的客户端发送给服务器一个请求, 这个请求将会被默默忽略.

共享密钥(shared secret)

客户端和服务器端也需要有一个共享的密钥, 将会用来加密和解密某些AVPs. `User-Password` AVP的值使用这个共享密钥加密. 当共享密钥在客户端和服务器端不同时, FreeRADIUS服务器将会检测到他并且当运行的debug模式时警告你.

```
Failed to authenticate the user.
    WARNING: Unprintable characters in the password. Double-check the shared secret on th
```

Message-Authenticator

当定义一个客户端你可以强制在所有请求中出现 `Message-Authenticator` AVP. 由于我们将会使用 `radtest` 程序, 没有包含他, 我们对于 `localhost` 禁用他, 通过设置 `require_message_authenticator` 为 `no` .

Nastype

`nastype` 设置为 `other` . `nastype` 的值将会剧而定 `checkrad` Perl脚本将会如何执行. `checkrad` 是用来决定一个用户是否已经使用一个NAS的资源. 由于 `localhost` 没有这个功能或者我们需要定义他为 `other` .

常见错误

如果服务器下线或者从 `radtest` 发出的包由于防火墙不能到达服务器, `radtest` 将会尝试3次然后以下面信息放弃

```
radclient: no response from server for ID 133 socket 3
```

如果你用普通用户运行 `radtest` , 他可能会抱怨没有权限访问FreeRADIUS字典文件. 这对于普通操作是需要的. 解决这个的方法是修改报告的文件的权限或者用root运行 `radtest`

users文件

用户定义在 `users` 文件在FreeRADIUS的配置目录. `users` 文件的内容是用来Authorization和Authentication. 这个文件不仅使用户的来源也是一个简单有效的方式开始. 让我们看一些关于users文件的关键点.

文件模块

`files` 模块(`rlm_files`)读取 `users` 文件的内容来决定是否在 `Access-Request` 中指定的用户存在并且授权来使用NAS. 他也决定了什么attribute应该被返回给客户端.

`files` 模块也设置了 `Auth-Type` . 这将会决定要使用的authentication方法. 如果一个用户定义了带有 `Cleartext-Password` check item, 他将会设置 `Auth-Type = PAP` .

`files` 模块也提供其他的模块带有一个值"know good password"如果定义了. 在我们的例子中, 他提供 `pap` 模块(`rlm_pap`)带有值指定在 `Cleartext-Password` check AVP.

PAP 模块

`pap` 模块(`rlm_pap`)是用来authentication. 如果 `Auth-Type` 设置为 `PAP` , 他将会寻找 `known good password` 并且将他跟 `User-Password` AVP的值比较. 如果是相同的, 这个模块将会通过authentication请求.

请求可能仍然失败由于在FreeRADIUS authentication 链条的其他模块.

Users文件

`users` 文件不包含任何section像 `clients.conf` . 这是因为他是针对 `files` 模块并且不直接跟FreeRADIUS服务器自身相关的.

要在 `users` 文件中添加一个入口, 你可以定义一个用户名跟着0个或者更多逗号分隔的检查项. 接着跟着0个或者更多tab缩进的行用逗号分隔的回复项.

我们假定你在使用默认的配置没有对 `sites-enabled/default` 虚拟服务器做任何修改. 如果你改了, 例如, 激活了在 `authorize` section下面的unix模块并且Alice也被定义为一个系统用户, 系统用户和他们密码将会是优先的用户而不是在 `users` 文件中定义的那个. 结果将会是一个 `Access-Reject` 包被返回来回应 `Access-Request` 如果在Alice(系统用户)和Alice(在 `user` 文件中用户)不同.

Check项目

下面的入口需要 `Access-Request` 包, 包含一个AVP对于 `NAS-IP-Address` 带有值 `127.0.0.1` .

```
"alice" Cleartext-Password := "passme", NAS-IP-Address == '127.0.0.1'
```

`radtest` 程序将会默认设置 `NAS-IP-Address` 的值为hostname的IP在 `/etc/hosts` 文件中指定的. 后面的章节中你将会看到这个值怎样修改.

一些AVPs有一个特殊的含义和被FreeRADIUS内部使用. 尽管进来的 `Access-Request` 没有包含一个AVP叫做 `Cleartext-Password` , 这个 `files` 模块内部使用他来调整 `Auth-Type` 的值并且来创建一个好的密码, 可以被 `pap` 模块使用来authentication.

另一个特殊的AVP的例子是当你想要拒绝或者接受一个基于用户的用户名, 无论他们的密码是什么. 之前的行将会改变为下面中的一个.

```
"alice" Cleartext-Password := "passme", Auth-Type := Reject
"alice" Cleartext-Password := "passme", Auth-Type := Accept
```

尽管 `Auth-Type` 似乎是一个标准的AVP检查项, 他属于FreeRADIUS内部, 并且是用来控制哪种authentication将会被使用.

回复项

开源项目有时因为缺少文档和支持被批判. FreeRADIUS在提供合适的文档跟提供帮助的方式上做了一个好的榜样.

安装的文档

有许多文档, 随着FreeRADIUS一起安装. 以manpages的形式, 在配置文件中的注释, 许多README文件, 和RFC文件.

Man pages

你可能不确定作为FreeRADIUS安装的一部分有哪些manpages. 下面一节将会展示给你如何你来发现这些.

下面的命令可以被用来指导首先决定哪些FreeRADIUS包被安装了和临时决定包中包含哪些文件.

dpkg系统

显示所有FreeRADIUS安装的包:

```
$> dpkg -l | grep radius
```

使用

术语表

FreeRADIUS

0. 介绍　1.1. 序言　1. ch00 序　2.2. RADIUS　2.3. FreeRADIUS　2.4. 总结　2. ch01 介绍 AAA 和 RADIUS　3.7. 编译 deb 包　3.3. 安装 FreeRADIUS　3.4. 从源码安装　3.5. 编译 RPM 包　3.6. 编译 SUSE 包　3.2. 预编译程序　3.8. 安装的二进制程序　3.9. 是否用 root 运行　3.10. 客户端程序的字典访问　3.11. 确保合适的启动　3.12. 总结　3. ch02 安装　4.1. 一个简单的配置　4.2. 配置 FreeRADIUS　4.3. 帮助你自己　4.4. 挖掘 FreeRADIUS 的 man page　4. ch03 开始使用 FreeRADIUS

RADIUS

Remote Authentication Dial In User Service

0. 介绍　1.1. 序言　2.1. Authentication, Authorization 和 Accounting　2.2. RADIUS　2.3. FreeRADIUS　2.4. 总结　2. ch01 介绍 AAA 和 RADIUS　3.3. 安装 FreeRADIUS　3.8. 安装的二进制程序　3.10. 客户端程序的字典访问　4.1. 一个简单的配置　4.2. 配置 FreeRADIUS　4. ch03 开始使用 FreeRADIUS